**Pintos Project 0-2: Pintos Data Structure**

1. **Additional Implementation**

|  |  |
| --- | --- |
| **Prototype** | int get\_bit\_idx(char \*) |
| **Parameter** | 비트맵 이름(문자열) |
| **Return** | 비트맵의 포인터가 있는 인덱스 |
| **Function** | 비트맵의 이름을 문자열로 받고 그 비트맵의 포인터가 있는 인덱스 값 반환 |

|  |  |
| --- | --- |
| **Prototype** | int get\_list\_idx(char \*) |
| **Parameter** | 리스트 이름(문자열) |
| **Return** | 리스트의 포인터가 있는 인덱스 |
| **Function** | 리스트의 이름을 문자열로 받고 그 리스트의 포인터가 있는 인덱스 값 반환 |

|  |  |
| --- | --- |
| **Prototype** | int get\_hash\_idx(char \*) |
| **Parameter** | 해시 이름(문자열) |
| **Return** | 해시의 포인터가 있는 인덱스 |
| **Function** | 해시의 이름을 문자열로 받고 그 해시의 포인터가 있는 인덱스 값 반환 |

|  |  |
| --- | --- |
| **Prototype** | void print\_bool(bool ) |
| **Parameter** | Boolean 값 |
| **Return** | 없음 |
| **Function** | true면 “true”출력, false면 “false”출력 |

|  |  |
| --- | --- |
| **Prototype** | bool get\_bool(char \*) |
| **Parameter** | 문자열 |
| **Return** | 문자열이 “true”면 true, “false”면 false |
| **Function** | 문자열이 “true”면 true, “false”면 false 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_manage(char \*) |
| **Parameter** | b로 시작하는 명령어(문자열) |
| **Return** | 없음 |
| **Function** | bitmap 명령어들을 받아서 처리 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_manage(char \*) |
| **Parameter** | h로 시작하는 명령어(문자열) |
| **Return** | 없음 |
| **Function** | hash 명령어들을 받아서 처리 |

|  |  |
| --- | --- |
| **Prototype** | void list\_manage(char \*) |
| **Parameter** | l로 시작하는 명령어(문자열) |
| **Return** | 없음 |
| **Function** | list 명령어들을 받아서 처리 |

|  |  |
| --- | --- |
| **Prototype** | bool list\_less(const struct list\_elem \*a, const struct list\_elem \*b, void \*aux) |
| **Parameter** | 비교할 리스트 항목 포인터 2개, 보조데이터 |
| **Return** | 첫번재 항목의 데이터가 두번째보다 작으면 true, 아니면 false |
| **Function** | 리스트 항목 두 개를 비교 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*get\_elem (struct list\*, int num) |
| **Parameter** | 리스트 포인터, 순서(정수) |
| **Return** | 리스트에서 num번째 항목 포인터 반환 |
| **Function** | 리스트에서 인자로 받은 양의 정수번째 오는 항목 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_func(const struct hash\_elem\* e, void\* aux) |
| **Parameter** | 해시 항목 포인터 |
| **Return** | 항목의 데이터의 해시값 |
| **Function** | 해시 항목의 데이터 값(정수)의 해시값 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_less(const struct hash\_elem\* a, const struct hash\_elem\* b, void\* aux) |
| **Parameter** | 비교할 해시 항목 포인터 2개, 보조데이터 |
| **Return** | 첫번재 항목의 데이터가 두번째보다 작으면 true, 아니면 false |
| **Function** | 해시 항목 두 개를 비교 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_square(struct hash\_elem \*e, void \*aux) |
| **Parameter** | 해시 항목 포인터 |
| **Return** | 없음 |
| **Function** | 해시 항목 포인터의 데이터 값을 제곱하여 저장 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_triple(struct hash\_elem \*e, void \*aux) |
| **Parameter** | 해시 항목 포인터 |
| **Return** | 없음 |
| **Function** | 해시 항목 포인터의 데이터 값을 세제곱하여 저장 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destructor(struct hash\_elem \*e, void \*aux) |
| **Parameter** | 해시 항목 포인터 |
| **Return** | 없음 |
| **Function** | 해시 항목이 속해있는 hash\_item을 free(메모리 해제) |

|  |  |
| --- | --- |
| **Prototype** | void list\_swap(struct list\_elem\* a, struct list\_elem\* b); |
| **Parameter** | 교환할 리스트 항목 포인터 두 개 |
| **Return** | 없음 |
| **Function** | 인자로 받은 항목의 데이터 값을 서로 교환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_shuffle(struct list\* list); |
| **Parameter** | 리스트 포인터 |
| **Return** | 없음 |
| **Function** | 리스트 내의 항목들의 순서를 랜덤으로 섞어 변경 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int\_2 (int); |
| **Parameter** | 정수 |
| **Return** | 정수의 해시값 |
| **Function** | 인자로 받은 정수의 해시값(10으로 나눈 나머지)를 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap\* bitmap\_expand(struct bitmap\* bitmap, int size); |
| **Parameter** | 비트맵 포인터, 크기 |
| **Return** | 확장된 비트맵 |
| **Function** | 인자로 받은 비트맵을 size크기만큼 확장하고 확장된 비트 값을 0으로 초기화한 뒤 반환 |

1. **List**

|  |  |
| --- | --- |
| **Prototype** | void ASSERT(CONDITION) |
| **Parameter** | 확인할 컨디션 |
| **Return** | 없음 |
| **Function** | assert(CONDITION)함수 호출, 컨디션이 true라면 아무 동작도 하지 않고 false라면 에러 메세지 출력 및 프로그램 종료 |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_head (struct list\_elem \*elem) |
| **Parameter** | 항목의 포인터 |
| **Return** | 항목이 head이면 true, 아니면 false |
| **Function** | 인자로 받은 항목의 head여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_interior (struct list\_elem \*elem) |
| **Parameter** | 항목의 포인터 |
| **Return** | 항목이 interior(head나 tail이 아닌 중간)이면 true, 아니면 false |
| **Function** | 인자로 받은 항목의 interior여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline bool is\_tail (struct list\_elem \*elem) |
| **Parameter** | 항목의 포인터 |
| **Return** | 항목이 tail이면 true, 아니면 false |
| **Function** | 인자로 받은 항목의 tail여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | list\_item \*list\_entry(list\_elem \*, STRUCT, MEMBER) |
| **Parameter** | 항목의 포인터 |
| **Return** | list\_elem의 포인터 |
| **Function** | 인자로 받은 항목이 속해있는 list\_item의 포인터 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_init (struct list \*); |
| **Parameter** | 초기화할 리스트의 포인터 |
| **Return** | 없음 |
| **Function** | 포인터가 가르키는 리스트를 빈 리스트로 초기화 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_begin (struct list \*); |
| **Parameter** | 첫 항목을 찾을 리스트의 포인터 |
| **Return** | 첫 항목의 포인터 |
| **Function** | 포인터가 가르키는 리스트의 첫 항목(head의 next) 반환 (list iterate에 사용) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_next (struct list\_elem \*); |
| **Parameter** | 리스트의 한 항목의 포인터 |
| **Return** | 다음 항목의 포인터 |
| **Function** | 인자로 받은 항목의 다음(next) 항목 반환 (list iterate에 사용) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_end (struct list \*); |
| **Parameter** | 끝을 찾을 리스트의 포인터 |
| **Return** | 끝(tail)의 포인터 |
| **Function** | 포인터가 가르키는 리스트의 끝(tail)을 반환 (list iterate에 사용) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rbegin (struct list \*); |
| **Parameter** | 마지막 항목을 찾을 리스트의 포인터 |
| **Return** | 마지막 항목의 포인터 |
| **Function** | 포인터가 가르키는 리스트의 마지막 항목(reverse beginning: tail의 prev) 반환 (list reverse iterate에 사용) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_prev (struct list\_elem \*); |
| **Parameter** | 리스트의 한 항목의 포인터 |
| **Return** | 이전 항목의 포인터 |
| **Function** | 인자로 받은 항목의 이전(prev) 항목 반환 (list iterate에 사용) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_rend (struct list \*); |
| **Parameter** | 시작을 찾을 리스트의 포인터 |
| **Return** | 시작(head)의 포인터 |
| **Function** | 포인터가 가르키는 리스트의 시작(reverse end: head)을 반환 (list reverse iterate에 사용) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_head (struct list \*); |
| **Parameter** | 시작을 찾을 리스트의 포인터 |
| **Return** | 시작(head)의 포인터 |
| **Function** | 포인터가 가르키는 리스트의 시작(head)을 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_tail (struct list \*); |
| **Parameter** | 끝을 찾을 리스트의 포인터 |
| **Return** | 끝(tail)의 포인터 |
| **Function** | 포인터가 가르키는 리스트의 끝(tail)을 반환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert (struct list\_elem \*, struct list\_elem \*); |
| **Parameter** | 삽입할 위치에 있는 항목의 포인터, 새 항목의 포인터 |
| **Return** | 없음 |
| **Function** | 두 번째 인자로 받은 항목을 첫 번째 인자 앞에 삽입 및 연결 |

|  |  |
| --- | --- |
| **Prototype** | void list\_splice (struct list\_elem \*before, struct list\_elem \*first, struct list\_elem \*last); |
| **Parameter** | 삽입할 위치에 있는 항목의 포인터, 삽입할 항목들의 시작과 끝 항목의 포인터 |
| **Return** | 없음 |
| **Function** | 시작과 끝 항목 이전까지 있는 모든 항목들을 순서 그대로 첫 인자로 받은 항목 앞에 삽입 및 연결 |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_front (struct list \*, struct list\_elem \*); |
| **Parameter** | 항목을 삽입할 리스트의 포인터, 삽입할 항목의 포인터 |
| **Return** | 없음 |
| **Function** | 리스트의 시작에 인자로 받은 항목을 삽입 및 연결 (첫 항목이 되도록) |

|  |  |
| --- | --- |
| **Prototype** | void list\_push\_back (struct list \*, struct list\_elem \*); |
| **Parameter** | 항목을 삽입할 리스트의 포인터, 삽입할 항목의 포인터 |
| **Return** | 없음 |
| **Function** | 리스트의 끝에 인자로 받은 항목을 삽입 및 연결 (마지막 항목이 되도록) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_remove (struct list\_elem \*); |
| **Parameter** | 리스트에서 제거할 항목의 포인터 |
| **Return** | 제거한 항목의 다음(next) 항목 |
| **Function** | 인자로 받은 항목을 그것이 속한 리스트에서 제거 및 앞 뒤 항목 연결 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_front (struct list \*); |
| **Parameter** | 항목을 제거할 리스트의 포인터 |
| **Return** | 제거한 항목의 포인터 |
| **Function** | 리스트의 첫 항목 제거 및 head와 뒤 항목 연결 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_pop\_back (struct list \*); |
| **Parameter** | 항목을 제거할 리스트의 포인터 |
| **Return** | 제거한 항목의 포인터 |
| **Function** | 리스트의 마지막 항목 제거 및 항목과 tail 연결 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_front (struct list \*); |
| **Parameter** | 항목을 찾을 리스트의 포인터 |
| **Return** | 첫 항목의 포인터 |
| **Function** | 리스트의 첫 항목 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_back (struct list \*); |
| **Parameter** | 항목을 찾을 리스트의 포인터 |
| **Return** | 마지막 항목의 포인터 |
| **Function** | 리스트의 마지막 항목 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t list\_size (struct list \*); |
| **Parameter** | 크기를 알고 싶은 리스트의 포인터 |
| **Return** | 리스트의 크기 |
| **Function** | 인자로 받은 리스트의 크기 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool list\_empty (struct list \*); |
| **Parameter** | 빈 리스트 여부를 알고 싶은 리스트의 포인터 |
| **Return** | 빈 리스트라면 true, 아니라면 false |
| **Function** | 인자로 받은 리스트의 빈 리스트 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | static void swap (struct list\_elem \*\*a, struct list\_elem \*\*b) |
| **Parameter** | 교환할 항목의 포인터의 포인터 |
| **Return** | 없음 |
| **Function** | 두 항목의 포인터 값을 서로 교환 |

|  |  |
| --- | --- |
| **Prototype** | void list\_reverse (struct list \*); |
| **Parameter** | 순서를 변경할 리스트의 포인터 |
| **Return** | 없음 |
| **Function** | 리스트의 항목의 순서를 반대로 변경 |

|  |  |
| --- | --- |
| **Prototype** | static bool is\_sorted (struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux) |
| **Parameter** | 시작 항목 포인터, 끝 항목 포인터, 리스트 비교함수, 보조데이터 |
| **Return** | a부터 b이전까지의 리스트 항목들이 정렬되어 있으면 true, 아니면 false |
| **Function** | a부터 b이전까지의 리스트 항목들이 정렬 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | static struct list\_elem \* find\_end\_of\_run (struct list\_elem \*a, struct list\_elem \*b, list\_less\_func \*less, void \*aux) |
| **Parameter** | 시작 항목 포인터, 끝 항목 포인터, 리스트 비교함수, 보조데이터 |
| **Return** | a부터 b까지의 리스트 항목들 중 정렬되지 않은 부분의 시작 포인터 |
| **Function** | a부터 시작해 b이전까지의 항목들 중 정렬되지 않은 부분의 시작(정렬된 부분의 다음 항목) 반환 |

|  |  |
| --- | --- |
| **Prototype** | static void  inplace\_merge (struct list\_elem \*a0, struct list\_elem \*a1b0, struct list\_elem \*b1, list\_less\_func \*less, void \*aux) |
| **Parameter** | 항목 포인터 3개, 리스트 비교함수, 보조데이터 |
| **Return** | 없음 |
| **Function** | a0 부터 a1b0 이전의 항목들의 순서를 변경하여 a0~b1 사이의 항목들이 비교함수에 따라 정렬될 수 있도록 처리, a0은 a1b0보다 앞이고 a1b0는 b1보다 앞이며, a1이 a1b0보다 크지 않을 때는 a0을 다음 항목으로 변경하고, 클 때는 a1b0를 다음 항목으로 변경하고 이전 a1b0의 값을 a0앞에 삽입(리스트 정렬에 사용) |

|  |  |
| --- | --- |
| **Prototype** | void list\_sort (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 정렬할 리스트의 포인터, 리스트 비교함수, 보조데이터 |
| **Return** | 없음 |
| **Function** | 리스트 비교함수에 따라 리스트를 오름차순으로 정렬 |

|  |  |
| --- | --- |
| **Prototype** | void list\_insert\_ordered (struct list \*, struct list\_elem \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 항목을 삽입할 리스트의 포인터, 삽입할 항목의 포인터, 리스트 비교함수, 보조데이터 |
| **Return** | 없음 |
| **Function** | 정렬된 리스트에 삽입할 원소의 위치를 비교함수에 따라 찾고, 그것을 삽입 및 연결 |

|  |  |
| --- | --- |
| **Prototype** | void list\_unique (struct list \*, struct list \*duplicates, list\_less\_func \*, void \*aux); |
| **Parameter** | 중복 항목을 제거할 리스트의 포인터, 제거하고 남은 항목을 삽입할 리스트의 포인터, 리스트 비교함수, 보조데이터 |
| **Return** | 없음 |
| **Function** | 첫 번째 인자로 받은 리스트에서 모든 값이 유일(unique)하도록 중복된 항목 제거, 제거하고 남은 항목을 두 번째 인자로 받은 리스트에 삽입(두 번째 인자가 NULL이면 실행 X) |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_max (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 항목을 찾을 리스트의 포인터, 리스트 비교함수, 보조데이터 |
| **Return** | 가장 큰 데이터 값을 가지는 항목의 포인터 |
| **Function** | 비교함수에 따라 리스트의 항목 중 가장 큰 데이터 값을 가진 항목 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct list\_elem \*list\_min (struct list \*, list\_less\_func \*, void \*aux); |
| **Parameter** | 항목을 찾을 리스트의 포인터, 리스트 비교함수, 보조데이터 |
| **Return** | 가장 작은 데이터 값을 가지는 항목의 포인터 |
| **Function** | 비교함수에 따라 리스트의 항목 중 가장 작은 데이터 값을 가진 항목 반환 |

1. **Hash Table**

|  |  |
| --- | --- |
| **Prototype** | hash\_elem\* list\_elem\_to\_hash\_elem(list\_elem \*) |
| **Parameter** | 리스트 항목의 포인터 |
| **Return** | 해시 항목의 포인터 |
| **Function** | 리스트 항목의 포인터가 속해있는 해시 항목의 포인터 |

|  |  |
| --- | --- |
| **Prototype** | hash\_item \*hash\_entry(hash\_elem \*, STRUCT, MEMBER) |
| **Parameter** | 항목의 포인터 |
| **Return** | hash\_item의 포인터 |
| **Function** | 인자로 받은 항목이 속해있는 hash\_item의 포인터 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_init (struct hash \*, hash\_hash\_func \*, hash\_less\_func \*, void \*aux); |
| **Parameter** | 초기화할 해시 포인터, 해시 함수, 해시 비교함수, 보조데이터 |
| **Return** | 초기화에 성공 시 true, 실패 시 false |
| **Function** | 인자로 받은 해시를 초기화 및 성공 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_clear (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 해시 포인터, 해시 수행 함수 |
| **Return** | 없음 |
| **Function** | 해시 항목 전체 삭제, 각 항목을 수행 함수에 따라 처리 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_destroy (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 제거할 해시의 포인터, 해시 수행 함수 |
| **Return** | 없음 |
| **Function** | 해시 항목 전체 삭제 및 해시 삭제(메모리 해제) |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_insert (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 항목을 삽입할 해시의 포인터, 삽입할 항목의 포인터 |
| **Return** | 기존에 항목이 존재했다면 그 항목의 포인터, 아니라면 NULL |
| **Function** | 인자로 받은 항목을 해시에 삽입(기존에 항목이 존재하면 실행X) |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_replace (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 항목을 삽입할 해시의 포인터, 삽입할 항목의 포인터 |
| **Return** | 기존에 항목이 존재했다면 그 항목의 포인터, 아니라면 NULL |
| **Function** | 인자로 받은 항목을 해시에 삽입(기존에 항목이 존재하면 교체) |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_find (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 항목을 찾을 해시의 포인터, 찾을 항목의 포인터 |
| **Return** | 항목을 찾았다면 그 항목의 포인터, 못찾았다면 NULL |
| **Function** | 해시 내부에 항목이 존재하는지 찾음 |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_delete (struct hash \*, struct hash\_elem \*); |
| **Parameter** | 항목을 제거할 해시 포인터, 제거할 항목의 포인터 |
| **Return** | 해시에 항목이 없다면 NULL, 있다면 제거한 항목 포인터 |
| **Function** | 해시 내부에 인자로 받은 항목을 찾고 삭제 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_apply (struct hash \*, hash\_action\_func \*); |
| **Parameter** | 해시 포인터, 해시 수행 함수 |
| **Return** | 없음 |
| **Function** | 해시 내부 모든 항목을 수행 함수에 따라 처리 |

|  |  |
| --- | --- |
| **Prototype** | void hash\_first (struct hash\_iterator \*, struct hash \*); |
| **Parameter** | 해시 반복자의 포인터, 해시 포인터 |
| **Return** | 없음 |
| **Function** | 해시 반복자가 해시의 첫 값을 가리키도록 초기화(hash iterate에 사용) |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_next (struct hash\_iterator \*); |
| **Parameter** | 해시 반복자의 포인터 |
| **Return** | 해시 반복자가 가르키던 항목의 다음 항목 포인터 |
| **Function** | 해시 반복자가 다음 값을 가리키도록 변경 및 다음 항목 반환(hash iterate에 사용) |

|  |  |
| --- | --- |
| **Prototype** | struct hash\_elem \*hash\_cur (struct hash\_iterator \*); |
| **Parameter** | 해시 반복자의 포인터 |
| **Return** | 반복자가 가르키는 항목의 포인터 |
| **Function** | 반복자가 가르키는 항목 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t hash\_size (struct hash \*); |
| **Parameter** | 해시 포인터 |
| **Return** | 해시의 사이즈 |
| **Function** | 해시의 사이즈(항목 개수) 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool hash\_empty (struct hash \*); |
| **Parameter** | 해시 포인터 |
| **Return** | 해시가 비었다면 true, 아니라면 false |
| **Function** | 인자로 받은 해시가 비었는지 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_bytes (const void \*, size\_t); |
| **Parameter** | 버퍼, 크기 |
| **Return** | 버퍼에 있는 크기만큼의 bytes의 해시값 |
| **Function** | 버퍼에 있는 크기만큼의 bytes의 해시값을 계산하여 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_string (const char \*); |
| **Parameter** | 문자열 |
| **Return** | 문자열의 해시값 |
| **Function** | 문자열의 해시값을 계산하여 반환 |

|  |  |
| --- | --- |
| **Prototype** | unsigned hash\_int (int); |
| **Parameter** | 정수 |
| **Return** | 정수의 해시값 |
| **Function** | 정수의 해시값을 계산하여 반환 |

|  |  |
| --- | --- |
| **Prototype** | static struct list \* find\_bucket (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 해시 포인터, 항목 포인터 |
| **Return** | 항목이 속한 해시 bucket(list)의 포인터 |
| **Function** | 항목이 속해있는 해시의 bucket 반환 |

|  |  |
| --- | --- |
| **Prototype** | static struct hash\_elem \* find\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | 해시 포인터, 해시 bucket(list) 포인터, 해시 항목 포인터 |
| **Return** | 항목이 bucket에 존재하면 그의 포인터, 아니면 NULL |
| **Function** | 해시 bucket내에서 인자로 받은 항목을 찾아 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t turn\_off\_least\_1bit (size\_t x) |
| **Parameter** | 변경할 값 x |
| **Return** | 변경한 x값 |
| **Function** | 값이 1인 x의 비트 중 가장 낮은 비트를 0으로 변경하고 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t is\_power\_of\_2 (size\_t x) |
| **Parameter** | 판단할 값인 x |
| **Return** | x가 2의 제곱수면 true, 아니면 false |
| **Function** | x의 2 제곱수 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | static void rehash (struct hash \*h) |
| **Parameter** | 해시 포인터 |
| **Return** | 없음 |
| **Function** | 해시의 bucket 크기를 현재 항목 개수에 알맞게 변경 |

|  |  |
| --- | --- |
| **Prototype** | static void insert\_elem (struct hash \*h, struct list \*bucket, struct hash\_elem \*e) |
| **Parameter** | 해시 포인터, 해시 bucket(list) 포인터, 해시 항목 포인터 |
| **Return** | 없음 |
| **Function** | 해시 bucket에 해시 항목 삽입 |

|  |  |
| --- | --- |
| **Prototype** | static void remove\_elem (struct hash \*h, struct hash\_elem \*e) |
| **Parameter** | 해시 포인터, 해시 항목 포인터 |
| **Return** | 없음 |
| **Function** | 해시 내부에서 인자로 받은 항목 삭제 |

1. **Bitmap**

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_idx (size\_t bit\_idx) |
| **Parameter** | bit\_idx값 |
| **Return** | bit\_idx의 비트를 포함하는 항목의 인덱스 값 |
| **Function** | bit\_idx의 비트를 포함하는 항목의 인덱스 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type bit\_mask (size\_t bit\_idx) |
| **Parameter** | bit\_idx값 |
| **Return** | elem type |
| **Function** | bit\_idx에 해당되는 비트가 1인 elem type반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t elem\_cnt (size\_t bit\_cnt) |
| **Parameter** | 비트 개수 bit\_cnt |
| **Return** | bit\_cnt개의 비트를 저장하기 위해 필요한 항목의 개수 |
| **Function** | bit\_cnt개의 비트를 저장하기 위해 필요한 항목의 개수 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline size\_t byte\_cnt (size\_t bit\_cnt) |
| **Parameter** | 비트 개수 bit\_cnt |
| **Return** | bit\_cnt개의 비트를 저장하기 위해 필요한 byte 개수 |
| **Function** | bit\_cnt개의 비트를 저장하기 위해 필요한 byte 개수 반환 |

|  |  |
| --- | --- |
| **Prototype** | static inline elem\_type last\_mask (const struct bitmap \*b) |
| **Parameter** | 비트맵 포인터 |
| **Return** | elem type |
| **Function** | 사용된 비트맵 항목의 비트는 1로 나머지는 0으로 설정하고 그elem type반환 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create (size\_t bit\_cnt); |
| **Parameter** | 비트 개수 |
| **Return** | 생성한 비트맵 포인터 |
| **Function** | 인자로 받은 비트 개수 크기의 비트맵 생성 및 반환 |

|  |  |
| --- | --- |
| **Prototype** | struct bitmap \*bitmap\_create\_in\_buf (size\_t bit\_cnt, void \*, size\_t byte\_cnt); |
| **Parameter** | 비트 개수, 메모리 공간, 메모리 공간의 크기(bytes) |
| **Return** | 생성한 비트맵 포인터 |
| **Function** | 인자로 받은 비트 개수 크기의 비트맵을 인자로 주어진 메모리공간에서 생성 및 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_buf\_size (size\_t bit\_cnt); |
| **Parameter** | 비트 개수 |
| **Return** | cnt 크기의 비트맵을 저장하기 위한 bytes 개수 |
| **Function** | 인자로 받은 비트 개수 크기의 비트맵을환저장하기 위한 bytes 개수 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_destroy (struct bitmap \*); |
| **Parameter** | 비트맵 포인터 |
| **Return** | 없음 |
| **Function** | 인자로 받은 비트맵 삭제(메모리 해제) |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_size (const struct bitmap \*); |
| **Parameter** | 비트맵 포인터 |
| **Return** | 비트맵의 크기 |
| **Function** | 인자로 받은 비트맵의 크기 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set (struct bitmap \*, size\_t idx, bool); |
| **Parameter** | 비트맵 포인터, 비트맵 인덱스, Boolean 값 |
| **Return** | 없음 |
| **Function** | 비트맵의 인덱스에 있는 값을 인자로 받은 Boolean 값으로 변경 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_mark (struct bitmap \*, size\_t idx); |
| **Parameter** | 비트맵 포인터, 비트맵 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵의 인덱스에 있는 값을 1으로 변경 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_reset (struct bitmap \*, size\_t idx); |
| **Parameter** | 비트맵 포인터, 비트맵 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵의 인덱스에 있는 값을 0으로 변경 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_flip (struct bitmap \*, size\_t idx); |
| **Parameter** | 비트맵 포인터, 비트맵 인덱스 |
| **Return** | 없음 |
| **Function** | 비트맵의 인덱스에 있는 값을 toggle(1이면 0으로 0이면 1로) |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_test (const struct bitmap \*, size\_t idx); |
| **Parameter** | 비트맵 포인터, 비트맵 인덱스 |
| **Return** | 해당 인덱스에 있는 비트 값(Boolean) |
| **Function** | 비트맵의 인덱스에 있는 값 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_all (struct bitmap \*, bool); |
| **Parameter** | 비트맵 포인터, Boolean 값 |
| **Return** | 없음 |
| **Function** | 비트맵의 모든 값을 인자로 받은 Boolean 값으로 변경 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_set\_multiple (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터, 시작 인덱스, 끝 인덱스, Boolean 값 |
| **Return** | 없음 |
| **Function** | 비트맵의 시작 인덱스와 끝 인덱스 사이의 값을 인자로 받은 Boolean 값으로 변경 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_count (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터, 시작 인덱스, 끝 인덱스, Boolean 값 |
| **Return** | 시작 인덱스~끝 인덱스의 비트 중 Boolean 값의 개수 |
| **Function** | 비트맵의 시작 인덱스와 끝 인덱스 사이의 값 중 인자로 받은 Boolean 값과 같은 값의 개수를 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_contains (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터, 시작 인덱스, 끝 인덱스, Boolean 값 |
| **Return** | 시작 인덱스~끝 인덱스의 비트 중 Boolean 값이 존재하면 true, 아니면 false |
| **Function** | 비트맵의 시작 인덱스와 끝 인덱스 사이의 값 중 인자로 받은 Boolean 값의 존재 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_any (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 비트맵 포인터, 시작 인덱스, 개수 |
| **Return** | start~(start+cnt-1)의 비트 중 1이 존재하면 true, 아니면 false |
| **Function** | 비트맵의 시작 인덱스부터 cnt개의 비트 중 1의 존재 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_none (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 비트맵 포인터, 시작 인덱스, 개수 |
| **Return** | start~(start+cnt-1)의 비트 중 1이 존재하면 false, 아니면 true |
| **Function** | 비트맵의 시작 인덱스부터 cnt개의 비트 중 1의 존재 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_all (const struct bitmap \*, size\_t start, size\_t cnt); |
| **Parameter** | 비트맵 포인터, 시작 인덱스, 개수 |
| **Return** | start~(start+cnt-1)의 비트가 모두 1이면 true, 아니면 flase |
| **Function** | 비트맵의 시작 인덱스부터 cnt개의 비트가 모두 1인지에 대한 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan (const struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터, 시작 인덱스, 개수, Boolean 값 |
| **Return** | start~(start+cnt-1)의 비트 중 cnt개의 연속된 Boolean 값을 갖는 인덱스 |
| **Function** | 비트맵의 시작 인덱스부터 cnt개의 비트 중 cnt개의 연속된 Boolean 값을 갖는 인덱스 반환 |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_scan\_and\_flip (struct bitmap \*, size\_t start, size\_t cnt, bool); |
| **Parameter** | 비트맵 포인터, 시작 인덱스, 개수, Boolean 값 |
| **Return** | start~(start+cnt-1)의 비트 중 cnt개의 연속된 Boolean 값을 갖는 인덱스 |
| **Function** | 비트맵의 시작 인덱스부터 cnt개의 비트 중 cnt개의 연속된 Boolean 값을 갖는 인덱스 반환하고 그 cnt개의 값을 toggle |

|  |  |
| --- | --- |
| **Prototype** | size\_t bitmap\_file\_size (const struct bitmap \*); |
| **Parameter** | 비트맵 포인터 |
| **Return** | 비트맵을 파일에 저장하기 위해 필요한 bytes개수 |
| **Function** | 비트맵을 파일에 저장하기 위해 필요한 bytes개수 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_read (struct bitmap \*, struct file \*); |
| **Parameter** | 비트맵 포인터, 파일 |
| **Return** | 파일을 읽는 데 성공했으면 true, 아니면 false |
| **Function** | 파일을 읽고 비트맵에 저장 및 성공 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | bool bitmap\_write (const struct bitmap \*, struct file \*); |
| **Parameter** | 비트맵 포인터, 파일 |
| **Return** | 파일 작성에 성공했으면 true, 아니면 false |
| **Function** | 비트맵을 파일에 작성 및 성공 여부 반환 |

|  |  |
| --- | --- |
| **Prototype** | void bitmap\_dump (const struct bitmap \*); |
| **Parameter** | 비트맵 포인터 |
| **Return** | 없음 |
| **Function** | 비트맵의 값을 16진수로 출력 |